REMARKS

Claims 1-35 are pending in the application, of which claims 1-34 stand rejected and claim 35 has been added to claim additional aspects of Applicant's invention.

REJECTIONS UNDER 35 U.S.C. 102

Claims 1-34 stand rejected under 35 U.S.C. 102(e) "as being anticipated by US Patent No. 6,144,965 to Oliver (hereinafter called Oliver)."

With regard to the rejection of claim 1, the Office Action states in relevant part that "Oliver disclose:... providing a function for automatically converting a smart pointer to an object of a first class to a smart pointer to an object of a second class... (col. 5, lines 12-22 'second entry... linked to each other')." Applicant respectfully disagrees.

The text of Oliver cited in the Office Action does not disclose Applicant's claimed feature of "providing a function for automatically converting a smart pointer to an object of a first class to a smart pointer to an object of a second class..." as recited in claim 1. The text of Oliver at column 5, lines 12-22, and as illustrated in Fig. 5B, illustrates what happens when the "object is copied" (Column 5, line 12), which is to create "a second entry in the pointer a list... for the same object." (Column 5, lines 13. Emphasis Added.) Oliver clearly states that the "second pointer list entry includes a second standard pointer to the _*original object*_." (Column 5, lines 14-15. Emphasis Added.) That is, Oliver explicitly indicates that both pointer list entries are _*standard pointers*_ that point to the _*same original object*_ without regard to the class(es) to which the object might belong. There is only one object disclosed at column 5, lines 12-22 of Oliver, and a single object not inherently an object of a first-class an object of a second-class. Therefore, the text at column 5, lines 12-22 fails to disclose "an object of a first-class" and "an object of a second-class" as recited in claim 1. Thus, Oliver fails to disclose Applicant's claimed features of "an object of a first class" and "an object of a second class" as recited in claim 1. Moreover, Oliver cannot therefore disclose Applicant's claimed features of "a smart pointer to an object of a first class" and "a smart pointer to an object of a second class" as recited in claim 1. Still further, Oliver makes no reference at column 5, lines 12-22 to providing a "function for converting..." as recited in claim 1. Consequently, for all the above reasons, Oliver fails to disclose at least Applicant's claimed feature of "providing a function for automatically converting a smart pointer to an object of a first class to a smart pointer to an object of a second class..." as recited in claim 1. Since, Oliver fails to disclose each and every element recited in claim 1, Applicant respectfully requests that the Examiner withdraw the rejection to claim 1, as well as claim 2 which depends from claim 1. However, yet additional reasons exist for withdrawing the rejection of claim 2.

With regard to the rejection of claim 2, the Office Action states that "Oliver disclose:... providing single member test for determining if a selected smart pointer is the only member of the ring and providing a deletion means for deleting the object of the selected smart pointer is determined to be the only member of the rain (col. 5, lines 23-29 'To delete a pointer... unreferenced... be deleted')." Applicant respectfully disagrees. Rather than disclose a single member test, Oliver discloses a two member test, which can lead to disastrous results in software operation, i.e. software crashes. Indeed, the Oliver two member test as disclosed at column 5, lines 23-29 and Fig. 5D creates one of the problems Applicant's claimed invention seeks to solve, that of the "dangling pointer, where reference is made to an object after its memory has been the released to the system (i.e., the object is no longer available)." (Application, paragraph [0004].)

Oliver states in relevant part at column 5, lines 25-29 that "[w]hen the 'next pointer' pointer and the 'previous pointer' pointer once again point to the same ... list entry, then the next pointer that is deleted is the last pointer to the object, and the object is then deemed unreferenced and may be deleted." This quoted text from Oliver in fact specifically discloses a two member test; that is, when "the next pointer" and "the previous pointer" point to the same list entry, there can be two members in the list, *__not a single member__*, Oliver's assertions to the contrary notwithstanding. (Oliver incorrectly asserts that this test is for a single list entry as evidenced by the text elided from the above quote.) The reason that is that this is a two member test is as follows.

Applicant respectfully submits that when "the next pointer" and "the previous pointer" point to the same list entry there can be *__two pointers on the list__*, as clearly disclosed by Oliver in Fig. 5B of Oliver, to which Applicant respectfully directs the Examiner's attention. Referring then to Fig. 5B, the figure illustrates a two pointer list (cf. column 5, lines 12-13) with the two pointers each labeled "LISTPTR" located side-by-side, one on the left and one on the right. For ease of discussion, Applicant refers to the "LISTPTR" on the left as "pointer A" and the

"LISTPTR" on the left is "pointer B". Since the list is in the form of a ring, "pointer A" both precedes "pointer B" and follows "pointer B". That is, Fig. 5B shows that the "next" pointer after "pointer A" is "pointer B", as indicated by the arrow that extends from the word "next" within "pointer A" and terminates with an arrowhead touching "pointer B". Similarly, Fig. 5B shows that the "previous" pointer before "pointer A" is "pointer B", as indicated by the arrow that extends from the word "previous" within "pointer A" and terminates with an arrowhead touching "pointer B". Consequently, with this understanding, it is clear that "the next pointer" of "pointer A" points to "pointer B", and that "the previous pointer" of "pointer A" also points to "pointer B". Hence, it is clear that for a two member list "the 'next pointer' pointer and the 'previous pointer' pointer once again point to the same ... list entry...". Thus, the test referred to in Oliver at column 5, lines 25-29 and illustrated in Fig. 5D at step 502 "is next = previous" is satisfied when there are two elements in the list. At which point, Oliver teaches at step 504 Fig. 5D and the text at column 5, lines 22-25 and 38-41 at to delete the object when "the 'next pointer' pointer and the 'previous pointer' pointer once again point to the same ... list entry...". This is distinct from Applicant's claimed invention as recited in claim 2, which specifically recites "providing *single member* test for determining if a selected smart pointer is the only member of the ring ...". (Emphasis Added.) Moreover, Oliver's two member test actually leads to a problem sought to be solved by Applicant, that of the dangling pointer.

As disclosed, Oliver indicates that one can delete the object when one of the pointers illustrated in Fig. 5B is to be deleted for the reasons presented above. This is what step 502 of Fig. 5D states. If one were to follow the disclosure of Oliver, the object in Fig. 5B would be deleted along with one of the pointers, leaving the remaining pointer pointing to a nonexistent object (i.e. a dangling pointer), with disastrous results. Such a situation could lead to program crashes on the remaining pointer is accessed in the object is to longer available to the pointer.

Oliver may have used the word "single" at column 5, line 27, but it is clear that Oliver has disclosed a test that returns a "true" result when there are two members on the list. Thus, *Oliver fails to provide an enabling disclosure* for a test that correctly identifies when only a single member is on a list. It is well settled that prior art under 35 U.S.C. 102 must sufficiently describe the claimed invention to have placed the public in possession of it. *In re Sasse*, 207 U.S.P. Q. 107, 11 (CCPA 1980). Accordingly, even if the claimed invention were arguably disclosed in a printed publication (which Applicant denies for the reasons set forth above), that disclosure would not suffice as prior art if it were not enabling. *In re Borst*, 45 U.S.P.Q. 554, 557 (CCPA, 1965). *In re Donohue*, 226 USPQ 619, 621 (Fed. Cir. 1985). See also MPEP 2121.01: "The disclosure in an assertedly anticipating reference must provide an enabling disclosure of the desired subject matter; mere naming or description of the subject matter is insufficient.... *Elan Pharm., Inc. v. Mayo Foundation for Medical and Education Research*, 346 F.3d 1051, 1054, 68 USPQ2d 1373, 1376 (Fed. Cir. 2003)" (Emphasis Added.) In the instant case Oliver may have *named* a single member test (test for a "(single) list entry"), but Oliver fails to teach the public how to *practice* such a test. Oliver *fails to provide the required enabling disclosure*. The public did not have possession of such a test that returns a true results only when there is a single member in the list.

Hence, for all the above reasons Applicant respectfully submits that Oliver fails to disclose at least Applicant's claimed feature of "providing single member test for determining if a selected smart pointer is the only member of the ring and providing a deletion means for deleting the object if the selected smart pointer is determined to be the only member of the ring", as recited in claim 2. Accordingly, for at least these additional reasons Applicant respectfully requests that the Examiner withdraw the rejection of claim 2.

Applicant further respectfully submits that the above arguments relating to the allowability of claim 2 over Oliver also provide a basis for allowing independent claims 3, 13, and 23, for the following reasons.

Regarding claim 3, claim 3 recites "providing a comparison means *for comparing the value of the next pointer to the value of the memory location of the smart pointer in which the selected next pointer is included*, whereby a determination can be made if the ring contains more than one smart pointer." (Emphasis Added.) That is, claim 3 recites "comparing the value of the next pointer" to "the value of the memory location of the smart pointer" "in which the selected next pointer is included." In other words, the comparison recited in the quoted text of claim 3 determines whether the next pointer of a particular smart pointer point to the smart pointer itself, i.e. does the next pointer point to its own smart pointer, which is a correct test for whether there is only one pointer on a ring. For example, step 410 exemplifies this concept by the test "next = this", as illustrated in Applicant's Fig. 4. In contrast, as explained above, Oliver tests not whether "next = this", but rather whether "next" equals "previous". Thus, Oliver fails to disclose at least Applicant's claimed comparison means "for comparing the value of the next pointer to the value of the memory location of the smart pointer in which the selected next pointer is included", as recited in claim 3. Accordingly, since Oliver fails to disclose each and every element of claim 3, Applicant respectfully requests that the Examiner withdraw the rejection of claim 3, as well as claims 4-12 which depend respectively therefrom.

Regarding claim 13, claim 13 recites "providing a comparison means *for comparing the value of the memory location of the smart pointer to the value of the next-pointer of the smart pointer*, to provide a determination whether the linked list contains only the smart pointer" (Emphasis Added.) That is, claim 13 recites comparing "the value of the next-pointer of the smart pointer" to "the value of the memory location of the smart pointer". In other words, the comparison recited in the quoted text of claim 13 determines whether the next pointer of a particular smart pointer points to the smart pointer itself, i.e. does the pointer point to itself, which is a correct test for whether there is only one pointer on a ring. For example, step 410 exemplifies this concept by the test "next = this", as illustrated in Applicant's Fig. 4. In contrast, as explained above, Oliver tests not whether "next = this", but rather whether "next" equals "previous". Thus, Oliver fails to disclose at least Applicant's claimed comparison means "for comparing the value of the memory location of the smart pointer to the value of the next-pointer of the smart pointer", as recited in claim 13. Accordingly, since Oliver fails to disclose each and every element of claim 13, Applicant respectfully requests that the Examiner withdraw the rejection of claim 13, as well as claims 14-22 which depend respectively therefrom.

Regarding claim 23, claim 23 recites "comparing the *value of the memory location of a selected smart pointer* giving up its association with the memory-resident element *to the value of the next-pointer of the selected smart pointer*, to provide a determination whether the linked list contains only the selected smart pointer" (Emphasis Added.) That is, claim 23 recites comparing "the value of the memory location of a selected smart pointer" to "the value of the next-pointer of the selected smart pointer". In other words, the comparison recited in the quoted text of claim 23 determines whether the next pointer of a particular smart pointer points to the smart pointer itself, i.e. does the pointer point to itself, which is a correct test for whether there is only one pointer on a ring. For example, step 410 exemplifies this concept by the test "next

= this", as illustrated in Applicant's Fig. 4. In contrast, as explained above, Oliver tests not whether "next = this", but rather whether "next" equals "previous". Thus, Oliver fails to disclose at least Applicant's claimed comparison means for "comparing the value of the memory location of a selected smart pointer giving up its association with the memory-resident element to the value of the next-pointer of the selected smart pointer", as recited in claim 23. Accordingly, since Oliver fails to disclose each and every element of claim 23, Applicant respectfully requests that the Examiner withdraw the rejection of claim 23, as well as claims 25-34 which depend respectively therefrom.

In view of the foregoing amendments and remarks, it is believed that the claims in this application are now in condition for allowance. Early and favorable reconsideration is respectfully requested. The Examiner is invited to telephone the undersigned in the event that a telephone interview will advance prosecution of this application.

Respectfully submitted,

Niels Haun
PTO Reg. No. 48,488

DANN DORFMAN HERRELL & SKILLMAN
A Professional Corporation
1601 Market Street, Suite 2400
Philadelphia, PA 19103
Phone: (215) 563-4100
Fax: (215) 563-4044